EUROPEAN PATENT OFFICE

Patent Abstracts of Japan

PUBLICATION NUMBER : 05224976
PUBLICATION DATE : 03-09-93

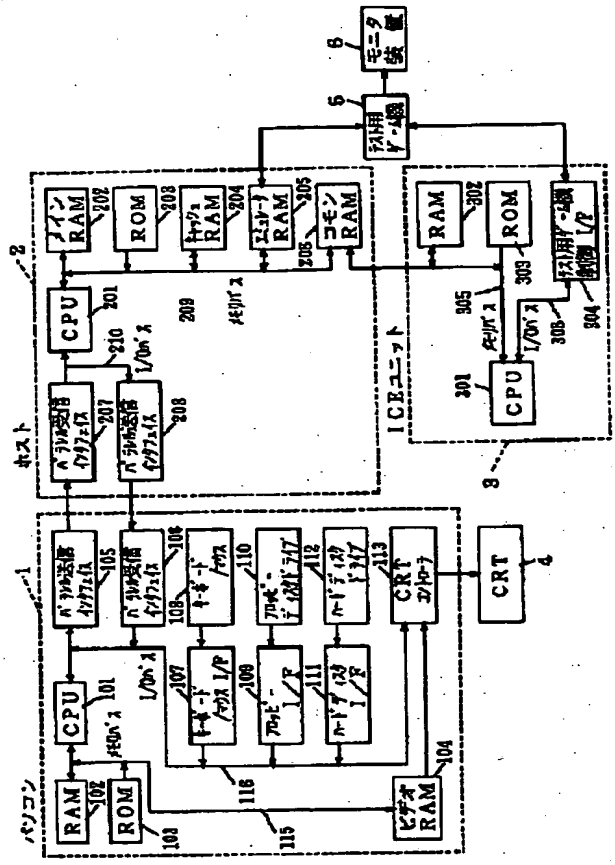
APPLICATION DATE : 10-02-92
APPLICATION NUMBER : 04057383

APPLICANT : NINTENDO CO LTD;

INVENTOR : WATANABE HIROYUKI;

INT.CL. : G06F 11/22

TITLE : PROGRAM DEVELOPMENT BACK-UP DEVICE



ABSTRACT : PURPOSE: To provide a program development back-up device which can develop the programs simply and in a short time.

CONSTITUTION: A program development back-up device is provided with a personal computer 1, a host computer 2, an ICE unit 3, and a CRT 4. The source program produced by the computer 1 is transferred to the computer 2 to undergo the assembly processing and the link processing and then converted into an object program. When both assembly and link processing are through via the computer 2, the object program is stored in an emulator RAM 205. Then the unit 3 actuates a test game machine 5 to carry out the program stored in the RAM 205. In such a constitution, the program is verified.

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平5－224976

(43)公開日　平成５年(1993)９月３日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　11/22	３４０　Ａ	8323－5B		

審査請求　未請求　請求項の数３（全 18 頁）

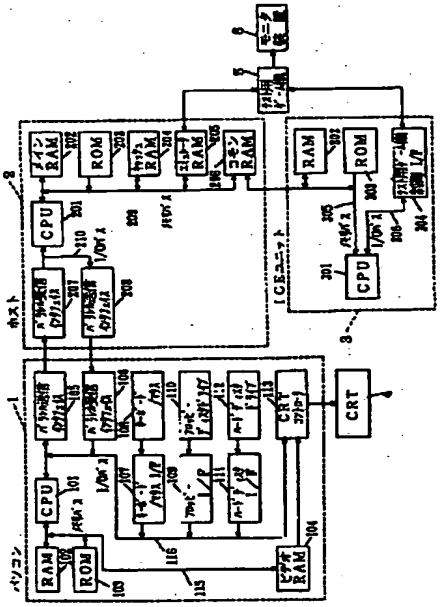
(21)出願番号　特願平4－57383

(22)出願日　平成４年(1992)２月10日

(71)出願人　000233778
任天堂株式会社
京都府京都市東山区福稲上高松町60番地
(72)発明者　富士本　淳
東京都大田区西馬込２丁目１番15号　株式会社セタ内
(72)発明者　渡辺　裕之
東京都大田区西馬込２丁目１番15号　株式会社セタ内
(74)代理人　弁理士　小笠原　史朗

(54)【発明の名称】　プログラム開発支援装置

(57)【要約】
【目的】　この発明の目的は、簡単にかつ短時間でプログラムの開発を行えるプログラム開発支援装置を提供することである。
【構成】　この発明のプログラム開発支援装置は、パソコン１と、ホストコンピュータ２と、ＩＣＥユニット３と、ＣＲＴ４とを備えている。パソコン１によって作成されたソースプログラムは、ホストコンピュータ２に転送されてアセンブル処理およびリンク処理が施され、オブジェクトプログラムに変換される。ホストコンピュータ２におけるアセンブルおよびリンク処理が終了すると、オブジェクトプログラムは、エミュレータＲＡＭ２０５に格納される。その後、ＩＣＥユニット３はテスト用ゲーム機５を動作させ、テスト用ゲーム機５にエミュレータＲＡＭ２０５内のプログラムを実行させる。これによって、プログラムの検証が行われる。



1

【特許請求の範囲】

【請求項1】　入力されたソースプログラムをオブジェクトプログラムに変換した後、当該オブジェクトプログラムをテスト装置において実行させることにより、作成中のプログラムを検証しながらプログラムの開発を行うプログラム開発支援装置であって、

第1のモニタ手段、

第1のデータ処理手段、および前記第1のデータ処理手段とデータ伝送可能に結合された第2のデータ処理手段を備え、

前記第1のデータ処理手段は、

ソースプログラムを入力するための入力手段と、

入力されたソースプログラムを前記第2のデータ処理装置へ送信するための送信手段と、

前記入力手段からの入力データを前記第1のモニタ手段に表示させるための表示制御手段とを含み、

前記第2のデータ処理手段は、

前記送信手段により送信されたソースプログラムを受信するための受信手段と、

前記受信手段により受信されたソースプログラムをアセンブルおよびリンクしてオブジェクトプログラムに変換するためのオブジェクトプログラム作成手段と、

前記オブジェクトプログラム作成手段により作成されたオブジェクトプログラムを記憶するエミュレータメモリと、

前記エミュレータメモリに記憶されたオブジェクトプログラムを前記テスト装置によって実行させるためのテスト装置制御手段とを含む、プログラム開発支援装置。

【請求項2】　前記第1のデータ処理手段は、前記入力手段から入力されたソースプログラムを記憶するソースプログラム記憶手段と、前記第2のデータ処理手段からのソースプログラムのアクセス要求に応答して、前記ソースプログラム記憶手段からソースプログラムを読み出して前記送信手段に送信させる送信制御手段とをさらに含み、

前記第2のデータ処理手段は、前記オブジェクトプログラム作成手段が作成中のオブジェクトプログラムを記憶するキャッシュメモリをさらに含み、

前記オブジェクトプログラム作成手段は、次に処理すべきデータが、前記キャッシュメモリに記憶されている場合は当該キャッシュメモリから該当のデータを直接ロードして処理し、前記キャッシュメモリに記憶されていない場合は前記第1のデータ処理手段に対して前記ソースプログラムのアクセス要求を発生する、請求項1に記載のプログラム開発支援装置。

【請求項3】　前記テスト装置は、

画像処理用プログラムを格納した外部メモリカートリッジが着脱自在に構成され、当該画像処理用プログラムに基づいて画像信号を発生する画像処理手段と、

前記画像処理手段に接続され、当該画像処理手段から与

2

えられる画像信号に基づく画像を発生する第2のモニタ手段とを含み、

前記テスト装置制御手段は、前記エミュレータメモリに記憶されているオブジェクトプログラムを前記画像処理用プログラムとして前記画像処理手段に与えることにより、作成中のプログラムに基づく画像を前記第2のモニタ手段に出力させる、請求項2に記載のプログラム開発支援装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】この発明は、プログラム開発支援装置に関し、より特定的には、入力されたソースプログラムをオブジェクトプログラムに変換した後、当該オブジェクトプログラムをテスト装置において実行させることにより、作成中のプログラムを検証しながらプログラムの開発を行うプログラム開発支援装置に関する。

【0002】

【従来の技術】テレビゲーム機（たとえば、本願出願人が販売する商品名「スーパーファミリーコンピューター」）のように所定のプログラムにしたがって動作するデータ処理装置が従来から知られている。従来、このようなデータ処理装置、たとえばテレビゲーム機に用いるプログラムの開発は、以下のようにして行われていた。

【0003】第1の方法は、汎用のパーソナルコンピュータ（以下、パソコンと略称する）で入力したソースプログラムを一旦パソコン内のワーキングＲＡＭに格納し、ソースプログラムの完成後にワーキングＲＡＭのデータをテレビゲーム機に適合する機械語に変換して外部のＲＯＭ基板に焼き付け、このＲＯＭ基板をテレビゲーム機に差し込んでチェックする方法である。

【0004】第2の方法は、汎用のパソコンで入力したソースプログラムを一旦パソコン内のワーキングＲＡＭに格納し、ソースプログラムの完成後にワーキングＲＡＭのデータをテレビゲーム機に適合する機械語に変換して外部のエミュレータＲＡＭに複製し、このエミュレータＲＡＭをテレビゲーム機に差し込んでチェックする方法である。

【0005】

【発明が解決しようとする課題】第1の方法は、作成されたゲームプログラムを修正する場合、新たなＲＯＭ基板を準備しなければならず、多大な時間と労力を必要とするという問題点があった。また、パソコンが種々の動作、すなわちデータの入力処理と、モニタの表示処理と、外部記憶装置（フロッピーディスクやハードディスク等）とのインタフェイスと、ＲＯＭ基板への書込制御と、プログラムデータのアセンブルおよびリンク処理と、テレビゲーム機のテスト動作の制御とを行わなければならず、パソコンの負荷が極めて大きくなる。その結果、パソコンの処理速度が遅くなり、プログラムの開発に長時間かかるという問題点もあった。

3

【0006】第2の方法は、作成されたゲームプログラムを修正する場合、同じエミュレータRAMに修正後のゲームプログラムをコピーし直せばよく、ROM基板を用いた第1の方法のような問題点を生じないが、依然としてパソコンの負荷は大きく、プログラムの開発に長時間を要する。

【0007】それゆえに、この発明の目的は、より簡単にかつ短時間でプログラムを作成し得るようなプログラム開発支援装置を提供することである。

【0008】

【課題を解決するための手段】請求項1に係る発明は、入力されたソースプログラムをオブジェクトプログラムに変換した後、このオブジェクトプログラムをテスト装置において実行させることにより、作成中のプログラムを検証しながらプログラムの開発を行うプログラム開発支援装置であって、第1のモニタ手段と、第1のデータ処理手段と、第1のデータ処理手段とデータ伝送可能に結合された第2のデータ処理手段とを備えている。第1のデータ処理手段は、ソースプログラムを入力するための入力手段と、入力されたソースプログラムを第2のデータ処理装置へ送信するための送信手段と、入力手段からの入力データを第1のモニタ手段に表示させるための表示制御手段と含んでいる。第2のデータ処理手段は、送信手段により送信されたソースプログラムを受信するための受信手段と、受信手段により受信されたソースプログラムをアセンブルおよびリンクしてオブジェクトプログラムに変換するためのオブジェクトプログラム作成手段と、オブジェクトプログラム作成手段により作成されたオブジェクトプログラムを記憶するエミュレータメモリと、エミュレータメモリに記憶されたオブジェクトプログラムをテスト装置によって実行させるためのテスト装置制御手段とを含む。

【0009】請求項2に係る発明では、第1のデータ処理手段は、入力手段から入力されたソースプログラムを記憶するソースプログラム記憶手段と、第2のデータ処理手段からのソースプログラムのアクセス要求に応答して、ソースプログラム記憶手段からソースプログラムを読み出して送信手段に送信させる送信制御手段とをさらに含んでいる。また、第2のデータ処理手段は、オブジェクトプログラム作成手段が作成中のオブジェクトプログラムを記憶するキャッシュメモリをさらに含んでいる。オブジェクトプログラム作成手段は、次に処理すべきデータが、キャッシュメモリに記憶されている場合はキャッシュメモリから該当のデータを直接ロードして処理し、キャッシュメモリに記憶されていない場合は第1のデータ処理手段に対してソースプログラムのアクセス要求を発生する。

【0010】請求項3に係る発明では、テスト装置は、画像処理手段と、第2のモニタ手段とを含んでいる。画像処理手段は、画像処理用プログラムを格納した外部メモリカートリッジが着脱自在に構成され、この画像処理用プログラムに基づいて画像信号を発生する。第2のモニタ手段は、画像処理手段に接続され、この画像処理手段から与えられる画像信号に基づく画像を発生する第2のモニタ手段とを含んでいる。テスト装置制御手段は、エミュレータメモリに記憶されているオブジェクトプログラムを画像処理用プログラムとして画像処理手段に与えることにより、作成中のプログラムに基づく画像を第2のモニタ手段に出力させる。

4

【0011】

【作用】請求項1に係るプログラム開発支援装置は、オブジェクトプログラム作成手段により作成されたオブジェクトプログラムをエミュレータメモリに格納し、このエミュレータメモリに格納されたオブジェクトプログラムを読み出してテスト装置に実行させるため、作成されたプログラムを修正する場合であってもエミュレータメモリの内容を書き換えるだけでよく、従来のように新たなメモリ基板を準備する必要がない。また、第1のデータ処理手段がデータの入力処理と、モニタの表示処理と、外部記憶装置（フロッピーディスクやハードディスク等）との間のインタフェイスとを行い、第2のデータ処理手段がエミュレータメモリへの書込制御と、プログラムデータのアセンブルおよびリンク処理と、テスト装置の動作制御とを行うため、負荷が分散され、プログラムの開発を短時間で行うことができる。

【0012】請求項2に係るプログラム開発支援装置においては、オブジェクトプログラム作成手段で作成中のオブジェクトプログラムをキャッシュメモリに記憶しておき、オブジェクトプログラム作成手段がアセンブルまたはリンク処理を実行する際に、まずキャッシュメモリをアクセスし、該当するデータがこのキャッシュメモリに格納されている場合は、オブジェクトプログラム作成手段がこのキャッシュメモリから直接データをロードしてデータを処理するようにしているため、データの高速アクセスが可能である。

【0013】請求項3に係るプログラム開発支援装置においては、テスト装置制御手段がエミュレータメモリに記憶されているオブジェクトプログラムを画像用プログラムとして画像処理手段に与えることにより、作成中のプログラムに基づく画像を第2のモニタ手段に出力させる。したがって、作成されたプログラムを即座に確認することができるため、プログラムの修正が容易である。

【0014】

【実施例】図1は、この発明の一実施例の構成を示すブロック図である。図において、この実施例のプログラム開発支援装置は、パソコン1と、ホストコンピュータ2と、ICE（In Circuit Emulata）ユニット3と、モニタ手段としてのCRT4とを備えている。

【0015】パソコン1は、画像表示のためのキャラク

タデータの入力およびキャラクタの表示座標または表示タイミングや種類等を指定するためのプログラムデータを入力する端末装置として用いられるものである。また、パソコン1は、本願発明をゲームプログラムの開発に適用する場合であれば、必要に応じて画像表示のためのプログラムの入力に加えて、ゲームの効果音またはゲーム音楽のプログラムを入力する端末装置としても用いられる。このパソコン1は、具体的にはCPU101と、RAM102と、ROM103と、ビデオRAM104と、パラレル送信インタフェイス105と、パラレル受信インタフェイス106と、キーボード/マウスインタフェイス107と、キーボード/マウス108と、フロッピーインタフェイス109と、フロッピーディスクドライブ110と、ハードディスクインタフェイス111と、ハードディスクドライブ112と、CRTコントローラ113とを含む。

【0016】RAM102，ROM103およびビデオRAM104は、メモリバス115を介してCPU101に接続されている。RAM102は、図2に示すように、記憶領域102a～102dを含む。記憶領域102aには、ホストコンピュータ2との間のデータ伝送のためのパソコン用パラレル送受信プログラムが格納されている。記憶領域102bには、パソコン側マクロコマンドを実行するためのパソコン側マクロコマンド実行プログラムが格納されている。記憶領域102cには、フロッピーディスクまたはハードディスクから読み出されたアプリケーションプログラムが格納されている。記憶領域102dは、CPU201の作業データを記憶する作業領域として用いられる。

【0017】ROM103には、図3に示すように、記憶領域103aを含む。この記憶領域103aには、IPL（Initial Program Loader）が格納されている。IPLは、パソコン1の起動時において、アプリケーションプログラムの初期プログラムを読み込む働きを有するプログラムである。

【0018】ビデオRAM104は、CRT4に表示すべき画像データを格納しておくためのメモリである。ビデオRAM104から読み出された画像データは、CRTコントローラ113に与えられる。

【0019】パラレル送信インタフェイス105，パラレル受信インタフェイス106，キーボード/マウスインタフェイス107，フロッピーインタフェイス109，ハードディスクインタフェイス111およびCRTコントローラ113は、入出力（以下、I/Oと略称する）バス116を介してCPU101に接続されている。

【0020】パラレル送信インタフェイス105は、CPU101の出力データをパラレルデータとしてホストコンピュータ2に送信するための回路であり、その送信出力はホストコンピュータ2におけるパラレル受信インタフェイス207に与えられる。パラレル受信インタフェイス106は、ホストコンピュータ2のパラレル送信インタフェイス208から送信されてくるパラレルデータを受信して、CPU101に出力するための回路である。キーボード/マウスインタフェイス107は、キーボード/マウス108からの入力データを取り込んでCPU101に出力するための回路である。フロッピーインタフェイス109は、フロッピーディスクドライブ110によってフロッピーディスク（図示せず）から読み取られたデータをCPU101に出力するための回路である。ハードディスクインタフェイス111は、ハードディスクドライブ112によってハードディスク（図示せず）から読み取られたデータをCPU101に出力するための回路である。

【0021】CRTコントローラ113は、CPU101から与えられる制御データおよびビデオRAM104から読み出された画像データに基づいて、CRT4に画像を表示させるための回路である。

【0022】ホストコンピュータ2は、CPU201と、メインRAM202と、ROM203と、キャッシュRAM204と、エミュレータRAM205と、コモンRAM206と、パラレル受信インタフェイス207と、パラレル送信インタフェイス208とを含む。

【0023】パラレル受信インタフェイス207およびパラレル送信インタフェイス208は、I/Oバス210を介してCPU201に接続されている。パラレル受信インタフェイス207は、パソコン1におけるパラレル送信インタフェイス105から送信されてくるパラレルデータを受信してCPU201に出力するための回路である。パラレル送信インタフェイス208は、CPU201から与えられるデータを、パラレルデータとしてパソコン1のパラレル受信インタフェイス207に送信するための回路である。

【0024】メインRAM202は、図4に示すように、記憶領域202a～202iを含む。記憶領域202aには、パソコン1から与えられるユーザーコマンドを照合し、実行するためのユーザーコマンド照合・実行プログラムプログラムが格納されている。記憶領域202bには、種々のユーザーコマンドデータを含むユーザーコマンドテーブルが格納されている。記憶領域202cには、ホスト側マクロコマンドを実行するためのホスト側マクロコマンド実行プログラムが格納されている。記憶領域202dには、ソースプログラムを中間言語に変換するための中間言語変換プログラムが格納されている。記憶領域202eには、アセンブル処理を実行するときに必要となる種々のオペレーションデータを含むアセンブルオペレーションテーブルが格納されている。記憶領域202fには、中間言語変換プログラムの実行により作成された中間言語データが格納されている。記憶領域202gには、アセンブル処理のためのアセンブル

実行プログラムが格納されている。記憶領域２０２ｈには、リンク処理のためのリンク実行プログラムが格納されている。記憶領域２０２１は、ＣＰＵ１０１の作業データを記憶する作業領域として用いられる。

【００２５】ＲＯＭ２０３は、図５に示すように、記憶領域２０３ａおよび２０３ｂを含む。記憶領域２０３ａには、パソコン１との間のデータ伝送のためのホスト用パラレル送受信プログラムが格納されている。記憶領域２０３ｂには、ＩＰＬが格納されている。

【００２６】キャッシュＲＡＭ２０４は、パソコン１から送信されてくるソースプログラムまたは作成中のオブジェクトプログラムを記憶する。エミュレータＲＡＭ２０５は、アセンブル処理およびリンク処理によって完成したオブジェクトプログラムを記憶する。ここで、アセンブル処理とは、アセンブラ言語によって作成されたソースプログラムを機械語（１か０かの数列）に変換することを言う。また、リンク処理とは、２つ以上のプログラムファイルを連結し、実行可能な形式にすることを言う。このリンク処理においては、たとえば主プログラムに対し、特定作業をするサブルーチンが連結されて実行可能なプログラムが作成される。また、ソースプログラムとは、プログラマがフローチャートを基にコンピュータに直接入力した最初のプログラムを言う。また、オブジェクトプログラムとは、ソースプログラムがアセンブラによって翻訳され、コンピュータが直接実行できる形（機械語）に変換されたプログラムを言う。エミュレータＲＡＭ２０５に格納された機械語としてのオブジェクトプログラムは、外部のテスト用ゲーム機５に与えられる。コモンＲＡＭ２０６は、ＩＣＥユニット３とのデータ通信用バッファとして用いられる。

【００２７】ＩＣＥユニット３は、ＣＰＵ３０１と、ＲＡＭ３０２と、ＲＯＭ３０３と、テスト用ゲーム機制御インタフェイス３０４とを含む。ＲＡＭ３０２およびＲＯＭ３０３は、メモリバス３０５を介してＣＰＵ３０１に接続されている。さらに、ＣＰＵ３０１には、メモリバス３０５を介してホストコンピュータ２におけるコモンＲＡＭ２０６が接続されている。テスト用ゲーム機制御インタフェイス３０４は、Ｉ／Ｏバス３０６を介してＣＰＵ３０１に接続されている。テスト用ゲーム機制御インタフェイス３０４は、さらに外部のテスト用ゲーム機５と接続されている。

【００２８】テスト用ゲーム機５は、作成されたオブジェクトプログラムをテストするためのゲーム機である。このテスト用ゲーム機５の構成は、任意であってよいが、この実施例では、本願出願人が販売する商品名「スーパーファミコン」または「スーパーＮｉｎｔｅｎｄｏ　ＥＮＴＥＲＴＡＩＮＭＥＮＴ　ＳＹＳＴＥＭ」を用いている。したがって、テスト用ゲーム機５は、ゲーム用プログラムが格納されたＲＯＭカートリッジを着脱自在に構成される。そして、テスト用ゲーム機５は、プログラムＲＯＭから読み出されたゲーム用プログラムに従ってＣＲＴ等のモニタ装置６に画像を表示させ、音声を出力させる。ホストコンピュータ２におけるエミュレータＲＡＭ２０５は、ＲＯＭカートリッジの代わりとして用いられ、そこから読み出されたプログラムがゲーム用プログラムとしてテスト用ゲーム機５に与えられる。したがって、テスト用ゲーム機５は、エミュレータＲＡＭ２０５から与えられるプログラムに従って画像信号および音声信号を発生し、モニタ装置６に与える。ＩＣＥユニット３は、テスト用ゲーム機５がプログラムのテストを行うときに、その動作を制御する。

【００２９】次に、図１に示す実施例の動作を説明する。図６および図７は、プログラムの作成と修正を行う場合におけるパソコン１とホストコンピュータ２の動作を関連的に示すフローチャートである。以下、この図６および図７を参照して、図１に示すプログラム開発支援装置の動作の全体的な流れを説明する。

【００３０】図１に示すプログラム開発支援装置の電源（図示せず）が投入されると、パソコン１側のＣＰＵ１０１は、ステップＳ１０１において、アプリケーションプログラムを起動する。このステップＳ１０１の動作は、ＲＯＭ１０３の記憶領域１０３ａ（図３参照）に記憶されたＩＰＬに従って実行される。すなわち、ＣＰＵ１０１は、フロッピーディスクドライブ１１０またはハードディスクドライブ１１２を駆動し、フロッピーディスクまたはハードディスク（図示せず）からパソコン用パラレル送受信プログラム，パソコン側マクロコマンド実行プログラムおよびアプリケーションプログラムを読み出して、それぞれ、ＲＡＭ１０２における記憶領域１０２ａ，１０２ｂおよび１０２ｃに格納する。以後、ＣＰＵ１０１は、ＲＡＭ１０２に記憶された各プログラムに従って動作を行う。

【００３１】一方、ホストコンピュータ２側のＣＰＵ２０１は、ステップＳ２０１において、初期設定を行う。このステップＳ２０１の初期設定は、ＲＯＭ２０３の記憶領域２０３ｂ（図４参照）に記憶されたＩＰＬに従って実行される。このとき、ＩＣＥユニット３も起動されている。

【００３２】次に、ＣＰＵ２０１は、ステップＳ２０２において、プログラムロード命令をパソコン１に転送する。このとき、プログラムロード命令は、Ｉ／Ｏバス２１０を介してパラレル送信インタフェイス２０８に与えられ、このパラレル送信インタフェイス２０８からパソコン１に送信される。パソコン１では、ホストコンピュータ２から送信されてきたプログラムロード命令をパラレル受信インタフェイス１０６によって受信する。パラレル受信インタフェイス１０６の受信データは、Ｉ／Ｏバス１１６を介してＣＰＵ１０１に与えられる。ＣＰＵ１０１は、与えられるプログラムロード命令に応答して、フロッピーディスクまたはハードディスクからホス

トコンピュータ2のためのプログラムデータを読み出し、パラレル送信インタフェイス105によってそれをホストコンピュータ2に送信する（ステップS102）。ホストコンピュータ2においては、パソコン1から送信されてきた各プログラムデータをパラレル受信インタフェイス207によって受信し、CPU201に与える。CPU201は、与えられた各プログラムデータを、メインRAM202における各記憶領域202a～202e，202g，202hにロードし、まず最初に記憶領域202cにロードされたホスト側マクロコマンド実行プログラムを起動する（ステップS203）。

【0033】次に、CPU201は、ステップS204において、1行入力命令をパソコン1に転送する。この1行入力命令は、ユーザーによってパソコン1に何らかのコマンドを入力させるための命令である。CPU101は、CPU201から与えられた1行入力命令に応答して、パソコン1をコマンド入力可能状態（いわゆるプロンプト状態）にする。このとき、ユーザーによってキーボードまたはマウスが操作され、キーボード／マウスインタフェイス107を介してCPU101にエディタ起動コマンドが入力される（ステップS103）。応じて、CPU101は、入力されたエディタ起動コマンドをホストコンピュータ2に転送する（ステップS104）。エディタ起動コマンドを受け取ったCPU201は、そのコマンドを解釈し、エディタ起動命令をパソコン1に転送する（ステップS205）。応じて、CPU101は、エディタを起動させる（ステップS105）。ここで、エディタとは、プログラムまたはデータをユーザーの指示に従って編集するツールを言い、主としてソースプログラムを作成する際に使用される。このエディタは、CPU101のソフト処理によって実現される。

【0034】次に、CPU101は、ユーザーがキーボードまたはマウスを操作することにより入力されたデータを、エディタによって編集し、ソースプログラムを作成する（ステップS106）。このとき作成されたソースプログラムは、一旦フロッピーディスクまたはハードディスクに格納される。なお、このソースプログラムの作成中は、CPU201は待機状態となっている（ステップS206）。ソースプログラムの作成が終了すると、CPU101はエディタを終了させ、その旨をホスト側コンピュータ2に転送する（ステップS107）。応じて、CPU201は、1行入力命令をパソコン1に転送する（ステップS207）。

【0035】次に、ユーザーは、キーボードまたはマウスを操作して、CPU101にアセンブル／リンクコマンドを入力する（ステップS108）。応じて、CPU101は、入力されたアセンブル／リンクコマンドをホストコンピュータ2に転送する（ステップS109）。

【0036】アセンブル／リンクコマンドを受け取ったCPU201は、アセンブル処理を実行する。すなわち、CPU201は、パソコン1によって作成されたソースプログラムを機械語に変換する。一方、CPU101は、ステップS110においてキーボードまたはマウスから入力されたエディタ起動コマンドに応答して、エディタを起動させている（ステップS111）。ホストコンピュータ2におけるアセンブル処理が終了すると（ステップS209）、CPU201は次にリンク処理を実行する（ステップS210）。このリンク処理によって、複数のプログラムファイルが連結され、全体として1つのオブジェクトプログラムにまとめられる。通常、ソースプログラムは、複数のブロックに分けて作成される。アセンブル処理では、各ブロック別にソースプログラムを機械語に変換する。リンク処理では、機械語に変換された各ブロックのデータを連結する。ホストコンピュータ2におけるリンク処理が終了すると（ステップS211）、CPU201はアセンブルおよびリンク処理の結果をパソコン1に転送する（ステップS212）。応じて、CPU101は、アセンブルおよびリンク処理の結果をCRT4に表示する（ステップS112）。

【0037】次に、CPU201は、1行入力命令をパソコン1に転送する（ステップS213）。このとき、ユーザーは、キーボードまたはマウスからエミュレータRAM転送コマンドおよびICEユニット動作コマンドをCPU101に入力する（ステップS113）。応じて、CPU101は、入力されたエミュレータRAM転送コマンドおよびICEユニット動作コマンドをホストコンピュータ2に転送する（ステップS114）。CPU201は、エミュレータRAM転送コマンドに応答して、キャッシュRAM204に格納されたオブジェクトプログラムをエミュレータRAM205に転送する（ステップS214）。

【0038】続いて、CPU201は、ICEユニット3にテスト動作開始の指示を与える（ステップS215）。応じて、ICEユニット3におけるCPU301は、テスト用ゲーム機制御インタフェイス304を介してテスト用ゲーム機5に起動命令を出力する。応じて、テスト用ゲーム機5は、エミュレータRAM205に格納されたオブジェクトプログラムを読み出し、それを実行する。このとき、ICEユニット3は、テスト用ゲーム機5の動作を制御する。たとえば、ICEユニット3は、テスト用ゲーム機5にリセット信号を与え、またデバッガのためのステップ動作用制御信号を与える。ステップ動作とは、プログラムの実行を、任意のタイミングで静止させるような動作を言う。

【0039】一方、パソコン1では、ユーザーによってキーボードまたはマウスからCPU101にエディタ起動コマンドが入力される（ステップS115）。応じて、CPU101は、エディタを起動させる（ステップ

S116）。次に、CPU101は、ユーザーによって入力されるデータをエディタを用いて編集し、作成されたソースプログラムの修正を行う（ステップS117）。このとき、ユーザーは、テスト用ゲーム機5におけるモニタ装置6の画像表示結果および音声出力結果に基づいて、ソースプログラム修正のためのデータを入力する。プログラムの修正が終了すると、CPU101はエディタを終了させる（ステップS118）。

【0040】CPU101はステップS118の動作が終了すると、ステップS108にリターンし、ステップS108以下の動作を繰り返す。CPU201は、ステップS215の動作が終了すると、ステップS207にリターンし、ステップS207以下の動作を繰り返す。したがって、修正されたソースプログラムに対して、再びアセンブルおよびリンク処理が施される。以後、同様の動作が繰り返され、作成されたプログラムに対して複数回修正作業がなされ、最終的なプログラムが完成する。なお、CPU201がリンク処理を行うことによって作成されるオブジェクトファイルは、キャッシュRAM204を介することなく、エミュレータRAM205に直接格納してもよい。これによって、キャッシュRAM204からエミュレータRAM205へのデータ転送の時間が省け、処理時間をさらに短縮することができる。

【0041】図8は、CPU201がパソコン1に対して1行入力命令を転送し、かつパソコン1から返送されてくるコマンドデータを解釈して実行する際の動作の詳細を示すフローチャートである。なお、この図8のフローチャートは、図6のステップS204，S205、またはステップS207，S208、または図7のステップS213，S214の動作に対応する。以下、この図8を参照して、CPU201による1行入力命令の転送動作およびそれに基づいて返送されてくるコマンドデータの処理動作について説明する。

【0042】まず、CPU201は、ステップS301において、1行入力処理を行う。このステップS301のサブルーチンの詳細は、図9のステップS401～S408に示されている。なお、図9は、パソコン1との対応関係を明確にするために、CPU101における対応する動作のフローチャートも示している。

【0043】図9を参照して、まずCPU201はパソコン受信フラグがオンされているか否かを判断する（ステップS401）。このパソコン受信フラグは、パソコン1がホストコンピュータ2からの送信データを受信可能か否かを示すものであり、パソコン1が受信可能状態に入る前にCPU101からCPU201に転送される。CPU201は、パソコン1からパソコン受信フラグが送信されたとき、パソコン受信フラグを内部のレジスタにロードし保持している。パソコン受信フラグがオンされている場合、すなわちパソコン1が受信可能状態の場合、CPU201は、ステップS402において、1行入力を指示するためのマクロコマンドをパソコン1に送信する。次に、CPU201は、ステップS403において、ホスト送信フラグをオンし、そのオンされたホスト送信フラグをパソコン1に送信する。

【0044】応じて、CPU101は、ホスト送信フラグがオンされていることを判断し（ステップS501）、ホストコンピュータ2から送信されてきた1行入力のためのマクロコマンドを入力する（ステップS502）。次に、ステップS503に進み、CPU101は、パソコン受信フラグをオンする。続いて、ステップS504に進み、CPU101は、1行入力のためのマクロコマンドを実行する。すなわち、CPU101は、コマンド入力可能状態（いわゆるプロンプト状態）であることをCRT4に表示させ、キーボードまたはマウスからのコマンド入力を受け付ける。

【0045】次に、ステップS505に進み、CPU101は、ホスト受信フラグがオンされているか否かを判断する。このとき、ホスト受信フラグはオンされているためステップS506に進み、CPU101は、マクロコマンドの実行結果、すなわちコマンドを表す文字データをホストコンピュータ2に送信する。続いて、ステップS507に進み、CPU101は、パソコン送信フラグをオンし、オンされたパソコン送信フラグをホストコンピュータ2に送信する。

【0046】応じて、CPU201は、パソコン送信フラグがオンされていることを判断する（ステップS404）。次に、ステップS405に進み、CPU201は、前述のステップS506においてパソコン1から送られてきたコマンドデータを入力する。次に、ステップS406に進み、CPU201は、ホスト受信フラグをオン状態にする。次に、ステップS407に進み、CPU201は、ステップS405で入力されたデータをメインRAM202の作業領域202iにストアする。次に、ステップS408に進み、CPU201は、入力データがキャリッジリターンコード（データの終了を示すコード）であるか否かを判断する。入力データがキャリッジリターンコードである場合は、再びステップS404～S408の動作が実行される。一方、入力データがキャリッジリターンコードである場合は、パソコン1からのデータ入力が終了したため、図8のステップS302にリターンする。

【0047】再び図8を参照して、CPU201は、前述のステップS407でメインRAM202の作業領域202iにストアされたコマンドデータを解釈する（ステップS302）。このステップS302の処理は、メインRAM202の記憶領域202aに格納されたユーザーコマンド照合・実行プログラムに基づいて行われる。すなわち、CPU201は、メインRAM202の作業領域202iにストアされたコマンドデータと、ユ

ーザーコマンドテーブル２０２ｂに予め格納されている複数種類のユーザーコマンドとを照合し、パソコン１から与えられたコマンドデータがいずれのユーザーコマンドと一致するかを検出する。

【００４８】次に、ステップＳ３０３に進み、ＣＰＵ２０１は、パソコン１から与えられたユーザーコマンドがホストコンピュータ２で処理すべきコマンドであるか、パソコン１で処理すべきコマンドであるかを判断する。ユーザーコマンドがホストコンピュータ２で処理すべきコマンドである場合は、ステップＳ３０４に進み、ＣＰＵ２０１は、そのユーザーコマンドに対応するプログラムを実行する。すなわち、メインＲＡＭ２０２の記憶領域２０２ｃから対応するホスト側マクロコマンド実行プログラムが選択されて実行される。ステップＳ３０４の動作の後、ＣＰＵ２０１は、動作を終了し、再び図６または図７の動作に戻る。

【００４９】一方、ユーザーコマンドがパソコン１で処理すべきコマンドである場合は、ステップＳ３０５に進み、ＣＰＵ２０１はパソコン受信フラグがオンされているか否かを判断する。パソコン受信フラグがオンされている場合、ステップＳ３０６に進み、ＣＰＵ２０１はユーザーコマンドに対応したマクロコマンドを出力し、パソコン１に送信する。このとき送信されるマクロコマンドは、メインＲＡＭ２０２の記憶領域２０２ａに格納されたユーザーコマンド照合・実行プログラムに基づいて発生される。次に、ステップＳ３０７に進み、ＣＰＵ２０１は、ホスト送信フラグをオン状態にする。続いて、ステップＳ３０８に進み、ＣＰＵ２０１は、パソコン送信フラグがオンされているか否かを判断する。パソコン送信フラグがオンされている場合、ステップＳ３０９に進み、ＣＰＵ２０１は、パソコン１から送信されてくるデータを入力する。次に、ステップＳ３１０に進み、ＣＰＵ２０１は、ホスト受信フラグをオン状態にする。以後、必要に応じて、ＣＰＵ２０１は、パソコン１に対して再度マクロコマンドを転送し、パソコン１からの返送データを入力する（ステップＳ３１１）。その後、ＣＰＵ２０１は動作を終了し、図６または図７の動作に戻る。

【００５０】図１０は、図６におけるステップＳ２０８～Ｓ２１１に対応するＣＰＵ２０１のより詳細な動作を示すフローチャートである。図１１および図１２は、図１０におけるステップＳ６０１のサブルーチンの詳細を示すフローチャートである。図１３は、図１０におけるステップＳ６０２のサブルーチンの詳細を示すフローチャートである。図１４は、図１０におけるステップＳ６０３のサブルーチンの詳細を示すフローチャートである。以下、これら図１０～図１４を参照して、ＣＰＵ２０１が実行するアセンブル処理およびリンク処理の詳細な動作を説明する。

【００５１】まず、図１０のステップＳ６０１において、ＣＰＵ２０１はアセンブル処理またはリンク処理の対象となるソースファイル（ソースプログラム中の１つのブロックを構成するデータ群）またはオブジェクトファイル（ソースファイルのデータをアセンブルしたデータ群）をロードする。次に、ステップＳ６０２に進み、ＣＰＵ２０１は、ロードしたソースファイルのデータを中間言語に変換する。ここで、中間言語は、ソースプログラム上の各命令を表す複数文字のデータを、それぞれに対応するコードに置き換えることにより、メモリ容量を節約する働きを持つ。このソースプログラムをコード化したものを中間言語と呼ぶ。次に、ステップＳ６０３に進み、ＣＰＵ２０１は、中間言語に変換されたソースプログラムをアセンブル、すなわちコンピュータが直接実行可能な機械語に変換する。

【００５２】次に、ステップＳ６０４に進み、ＣＰＵ２０１は、キャッシュＲＡＭ２０４内に格納された各オブジェクトファイルにリンク処理を施す。次に、ステップＳ６０５に進み、ＣＰＵ２０１は、リンク処理が終了したか否かを判断する。リンク処理が終了していなければ、再びステップＳ６０４、Ｓ６０５の動作が実行される。リンク処理が終了した場合は、図６の動作にリターンする。

【００５３】次に、図１１および図１２を参照して、図１０におけるステップＳ６０１のサブルーチンのより詳細な動作を説明する。まずステップＳ７０１においてＣＰＵ２０１は、キャッシュＲＡＭ２０４内にアセンブル処理またはリンク処理に必要なソースファイルまたはオブジェクトファイルが格納されているか否かを判断する。キャッシュＲＡＭ２０４内に該当するファイルが格納されていない場合は、ステップＳ７０２に進み、ＣＰＵ２０１はパソコン受信フラグがオンされているか否かを判断する。パソコン受信フラグがオンされている場合は、ステップＳ７０３に進み、ＣＰＵ２０１はソースプログラムをロードするためのマクロコマンドを出力し、パソコン１に送信する。次に、ステップＳ７０４に進み、ＣＰＵ２０１はパソコン受信フラグがオンされているか否かを判断する。パソコン受信フラグがオンされている場合は、ソースプログラムのロードに必要なデータおよびパラメータを１バイトずつ出力し、パソコン１に送信する。次に、ステップＳ７０６に進み、ＣＰＵ２０１はホスト送信フラグをオン状態にする。次に、ステップＳ７０７に進み、ＣＰＵ２０１はパソコン１へのデータの出力が終了したか否かを判断する。データの出力が終了していない場合は再びステップＳ７０４の動作に戻り、ステップＳ７０４～Ｓ７０７の動作が繰り返される。

【００５４】パソコン１へのデータの出力が終了すると、図１２のステップＳ７０８に進み、ＣＰＵ２０１はパソコン送信フラグがオンされているか否かを判断する。このとき、パソコン１は、フロッピーディスクまた

15

はハードディスクからソースプログラムを読み出して、1バイトずつホストコンピュータ2に送信する。パソコン送信フラグがオンされていると、ステップS709に進み、CPU201はパソコン1から1バイトずつ送られてくるソースプログラムデータを入力し、キャッシュRAM204に格納する。次に、ステップS710に進み、CPU201はホスト受信フラグをオン状態にする。次に、ステップS711に進み、CPU201はパソコン1から送信されてくるソースプログラムデータが終了したか否かを判断する。送信データが終了していない場合は、再びステップS708の動作に戻り、前述のステップS708～S711の動作が繰り返される。

【0055】一方、送信データが終了すると、ステップS712に進み、CPU201はキャッシュディレクトリを作成する。キャッシュRAM204の中には、ディレクトリ領域とデータ領域とがある。データ領域には、ソースプログラムの各ファイルが格納される。ディレクトリ領域には、各ファイルごとにファイル名と、サイズと、データ作成の日付と、データ領域の先頭アドレスとが格納されている。ステップS712では、データ領域における各ファイルのディレクトリデータが作成されてキャッシュRAM204のディレクトリ領域に格納される。次に、ステップS713に進み、CPU201はキャッシュRAM204に格納されている各ファイルの先頭アドレスをパラメータとして内部レジスタにロードする。

【0056】一方、前述のステップS701においてキャッシュRAM204の内部に該当するファイルが格納されている場合は、CPU201はステップS702～S712の動作をスキップして、直接ステップS713の動作を実行する。すなわち、CPU201は、パソコン1からデータを転送することなくキャッシュRAM204における各ファイルの先頭アドレスをパラメータとしてロードする。

【0057】次に、図13を参照して、図10に示すステップS602のサブルーチンのより詳細な動作を説明する。まず、CPU201は、ステップS801においてキャッシュRAM204から最初のソースファイルを読み出す。この読出動作は、前述のステップS713（図12参照）においてCPU201の内部レジスタにロードされた各ソースファイルの先頭アドレスを参照して行われる。次に、ステップS802に進み、CPU201はメインRAM202の記憶領域202eに格納されたアセンブルオペレーションテーブルを参照する。次に、ステップS803に進み、CPU201は、ステップS801で読み出したソースファイルのデータを中間言語データに変換し、メインRAM202の記憶領域202fに格納する。このステップS803における中間言語データへの変換処理は、ステップS802で参照したアセンブルオペレーションテーブルのアセンブルオペ

16

レーションコードに基づいて行われる。次にステップS804に進み、CPU201はキャッシュRAM204中のすべてのソースファイルを中間言語に変換したか否かを判断する。中間言語に変換すべきソースファイルが残っている場合は、再びステップS801～S804の動作が繰り返される。一方、すべてのソースファイルの変換が終了した場合は、図10の動作にリターンする。

【0058】次に、図14を参照して、図10に示すステップS603のサブルーチンのより詳細な動作を説明する。まず、CPU201はステップS901において、メインRAM202の記憶領域202fから中間言語に変換されたソースファイルデータを読み出す。次に、ステップS902に進み、CPU201はソースファイル中の命令数を演算することにより、所定の命令に付与された命令ラベルの相対アドレス値を演算する。次に、ステップS903に進み、CPU201はキャッシュRAM204内のすべてのソースファイルに対して相対アドレス値の演算が終了したか否かを判断する。相対アドレス値の演算が終了していない場合は、再びステップS901～S903の動作が繰り返される。

【0059】一方、すべてのソースファイルに対する相対アドレス値の演算が終了した場合は、ステップS904に進み、CPU201はメインRAM202の記憶領域202fから中間言語に変換されたソースファイルを読み出す。次に、ステップS905に進み、CPU201は中間言語に変換されたソースファイルをアセンブルすなわち機械語に変換し、オブジェクトファイルとして、キャッシュRAM204にストアする。次に、ステップS906に進み、CPU201は、キャッシュRAM204内のすべてのソースファイルに対するアセンブル処理が終了したか否かを判断する。アセンブル処理を行うべきソースファイルが残っている場合は、再びステップS904～S906の動作が繰り返される。一方、すべてのソースファイルに対するアセンブル処理が終了した場合は、図10のステップS604の動作にリターンする。

【0060】次に、本実施例におけるキャッシュRAM204の働きおよび利点をより詳細に説明する。

【0061】まず、ソースプログラムのアセンブル処理においては、パソコン1からホストコンピュータ2に順次ソースファイルが転送されてキャッシュRAM204に格納される。CPU201は、キャッシュRAM204に格納された各ソースファイルに対してアセンブル処理を行なう。次に、リンク処理を行う場合は、すでにキャッシュRAM204内に各オブジェクトファイルが格納されているため、CPU201はキャッシュRAM204から直接各オブジェクトファイルをロードしてリンク処理を実行することができる。したがって、パソコン1からデータを転送する必要がないので、データのアクセス時間を短縮化することができる。次に、試作された

17

プログラムのいずれかの部分を修正する場合は、パソコン１から該当するソースファイル（修正されたソースファイル）のみが入力され、ホストコンピュータ２に転送される。その他のソースファイルについてはキャッシュＲＡＭ２０４に格納されているので、ＣＰＵ２０１はパソコン１に対して各ファイルの転送を指令する必要がない。したがって、データ転送の時間を節約できる。さらに、修正されたソースファイルをアセンブルして得られたオブジェクトファイルのリンク処理についても、キャッシュＲＡＭ２０４にはすでに他のオブジェクトファイルがリンクされた形態で格納されているため、修正されたオブジェクトファイルについてのみリンク処理を行えばよい。したがって、リンク処理時間の短縮化を図ることができる。

【００６２】

【発明の効果】以上のように、この発明によれば、第１および第２のデータ処理手段が連携して画像表示のためのプログラムの作成作業を行うとともに、作成したソースプログラムを機械語に変換して直ちにテスト装置に表示しているため、１つのパソコンが全ての処理を行う従来のプログラム開発装置に比べて、負荷が分散され、各データ処理装置はそれぞれの処理を高速で行うことができる。したがって、プログラムの開発が短時間で行える。

【００６３】また、請求項２にかかる発明によれば、オブジェクトプログラム作成手段がアセンブルまたはリンク処理を実行する際に、まずキャッシュメモリをアクセスし、該当するデータがこのキャッシュメモリに格納されている場合は、オブジェクトプログラム作成手段がこのキャッシュメモリから直接データをロードしてデータを処理するようにしているため、データの高速アクセスが可能となり、プログラムの処理速度を向上することができる。

【００６４】請求項３にかかる発明によれば、エミュレータメモリに記憶されているソースプログラムデータをテスト装置の画像処理手段に与えて、作成中のソースプログラムデータに基づく画像を第２のモニタ手段に出力させるようにしているので、作成されたソースプログラムデータを即座に確認することができる。したがって、プログラムの修正が容易である。

【図面の簡単な説明】

【図１】この発明の一実施例の構成を示すブロック図である。

【図２】図１におけるパソコン内に設けられたＲＡＭのメモリマップを示す図解図である。

18

【図３】図１におけるパソコン内に設けられたＲＯＭのメモリマップを示す図解図である。

【図４】図１におけるホストコンピュータ内に設けられたメインＲＡＭのメモリマップを示す図解図である。

【図５】図１におけるホストコンピュータ内に設けられたＲＯＭのメモリマップを示す図解図である。

【図６】図１に示す実施例がプログラムを作成および修正する際の全体的な動作の流れを、パソコンおよびホストコンピュータのそれぞれについて示すフローチャートである。

【図７】図６に示すパソコンおよびホストコンピュータの動作の続きを示すフローチャートである。

【図８】図１に示すホストコンピュータがパソコンに対して１行入力命令を転送し、かつパソコンからの返送コマンドデータを処理する場合の動作を示すフローチャートである。

【図９】図８に示すステップＳ３０１のサブルーチンのより詳細な動作およびそれに対応するパソコンの動作を示すフローチャートである。

【図１０】図１に示すホストコンピュータがアセンブルおよびリンク処理を実行する際の動作を示すフローチャートである。

【図１１】図１０におけるステップＳ６０１のサブルーチンのより詳細な動作を示すフローチャートである。

【図１２】図１１に示すサブルーチンの動作の続きを示すフローチャートである。

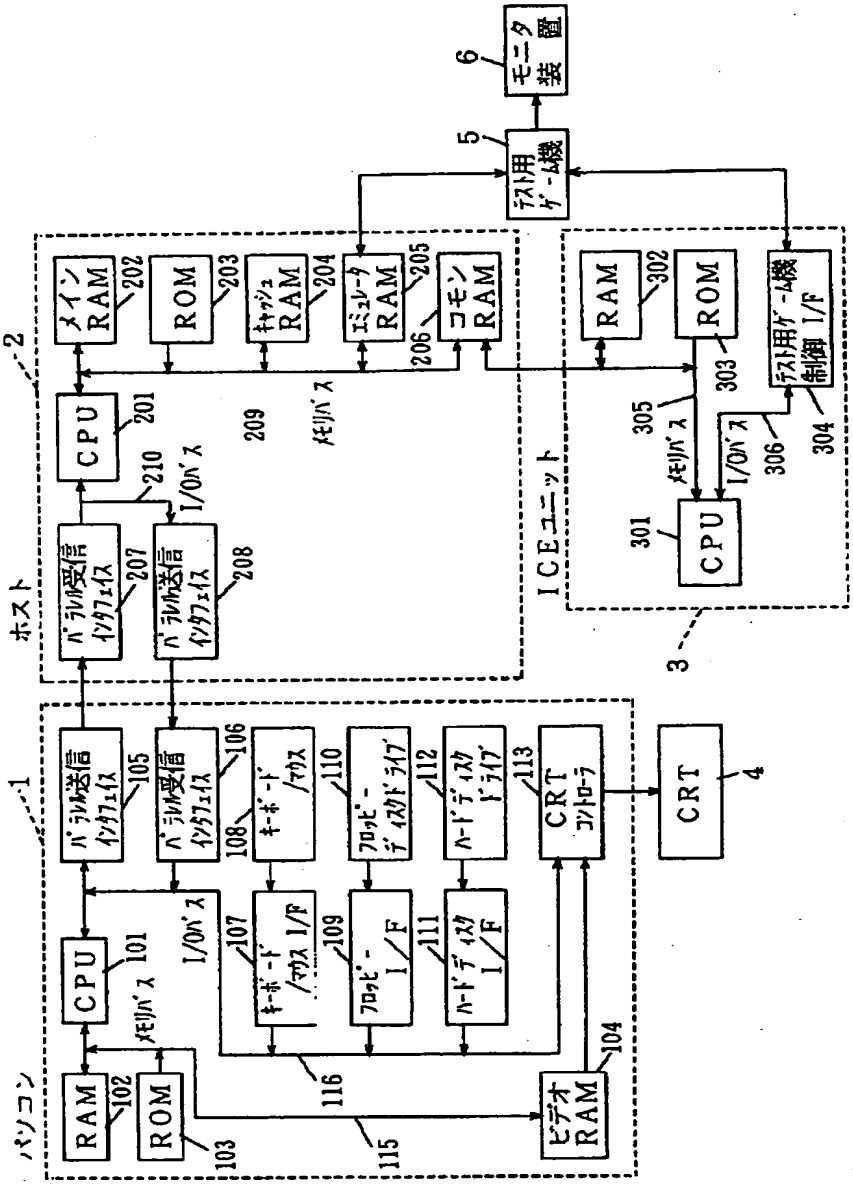
【図１３】図１０におけるステップＳ６０２のサブルーチンのより詳細な動作を示すフローチャートである。

【図１４】図１０におけるステップＳ６０３のサブルーチンのより詳細な動作を示すフローチャートである。

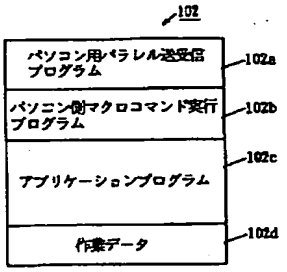
【符号の説明】

１： パーソナルコンピュータ
２： ホストコンピュータ
３： ＩＣＥユニット
４： ＣＲＴ
５： テスト用ゲーム機
１０１，２０１，３０１： ＣＰＵ
１０２，３０２： ＲＡＭ
１０３，２０３，３０３： ＲＯＭ
２０２： メインＲＡＭ
２０４： キャッシュＲＡＭ
２０５： エミュレータＲＡＭ
１０５，２０８： パラレル送信インタフェイス
１０６，２０７： パラレル受信インタフェイス
１１３： ＣＲＴコントローラ

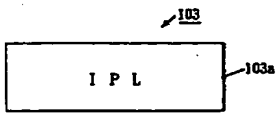
【図1】



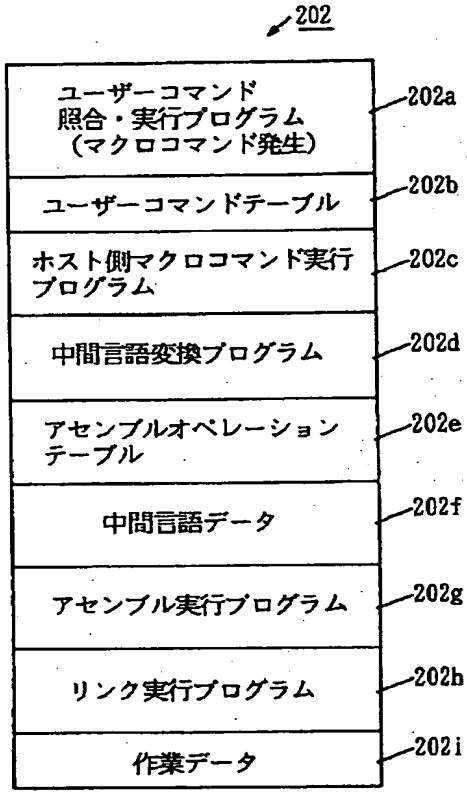
【図２】



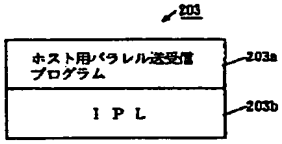
【図３】



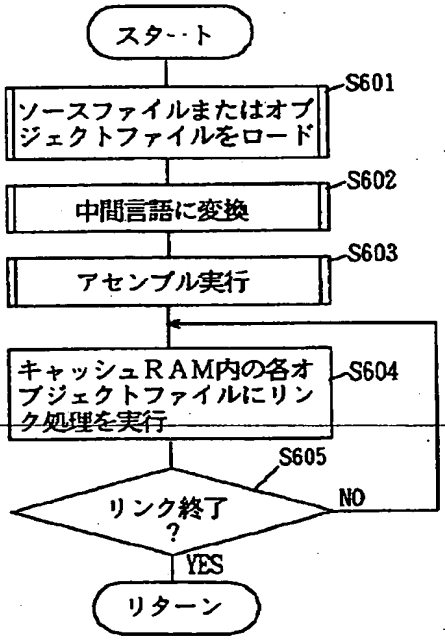
【図４】



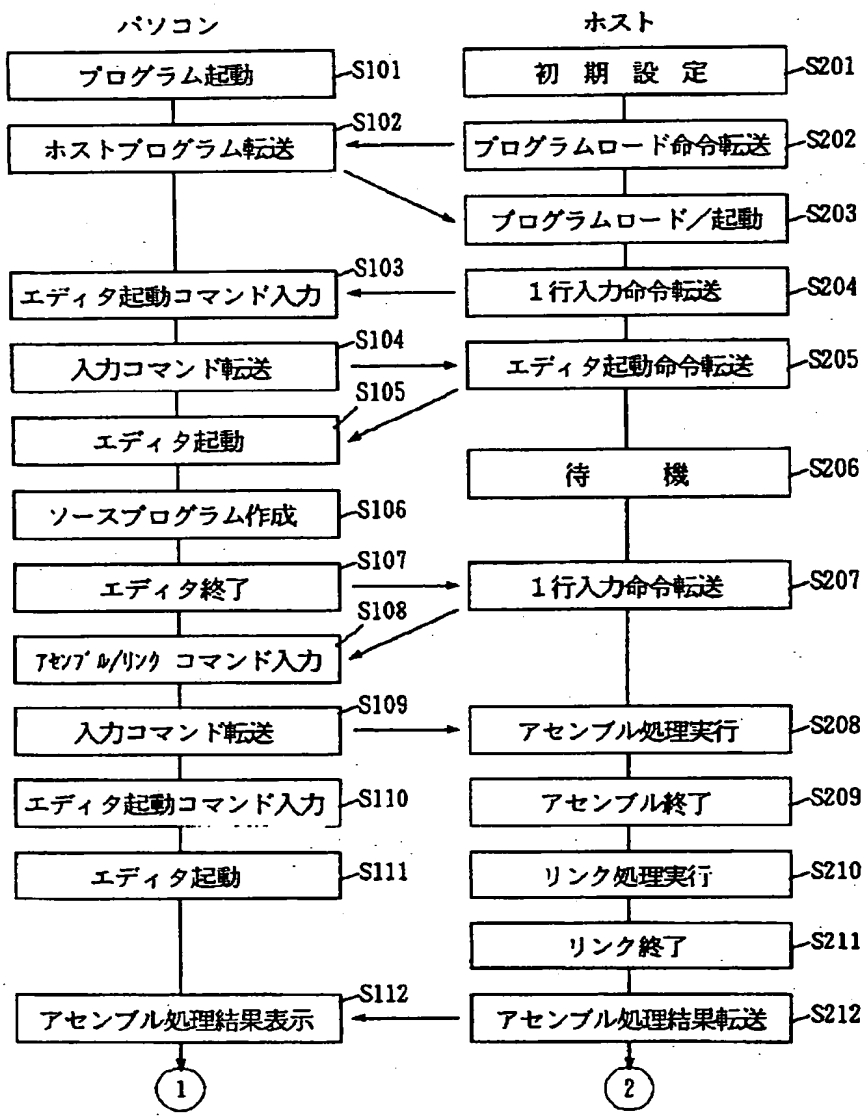
【図５】



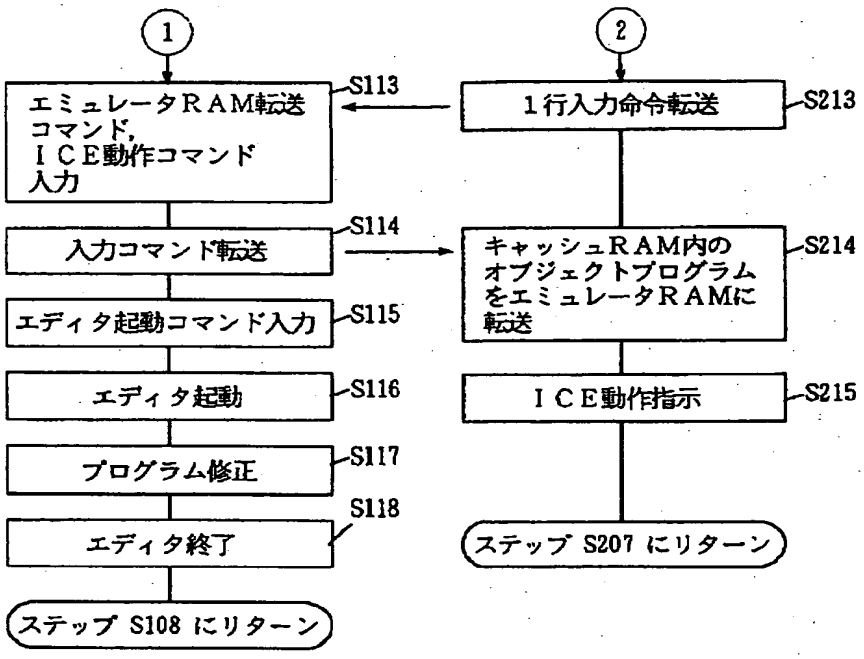
【図１０】



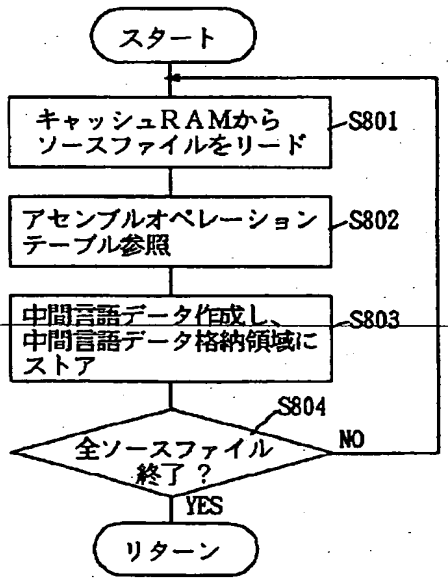
【図6】



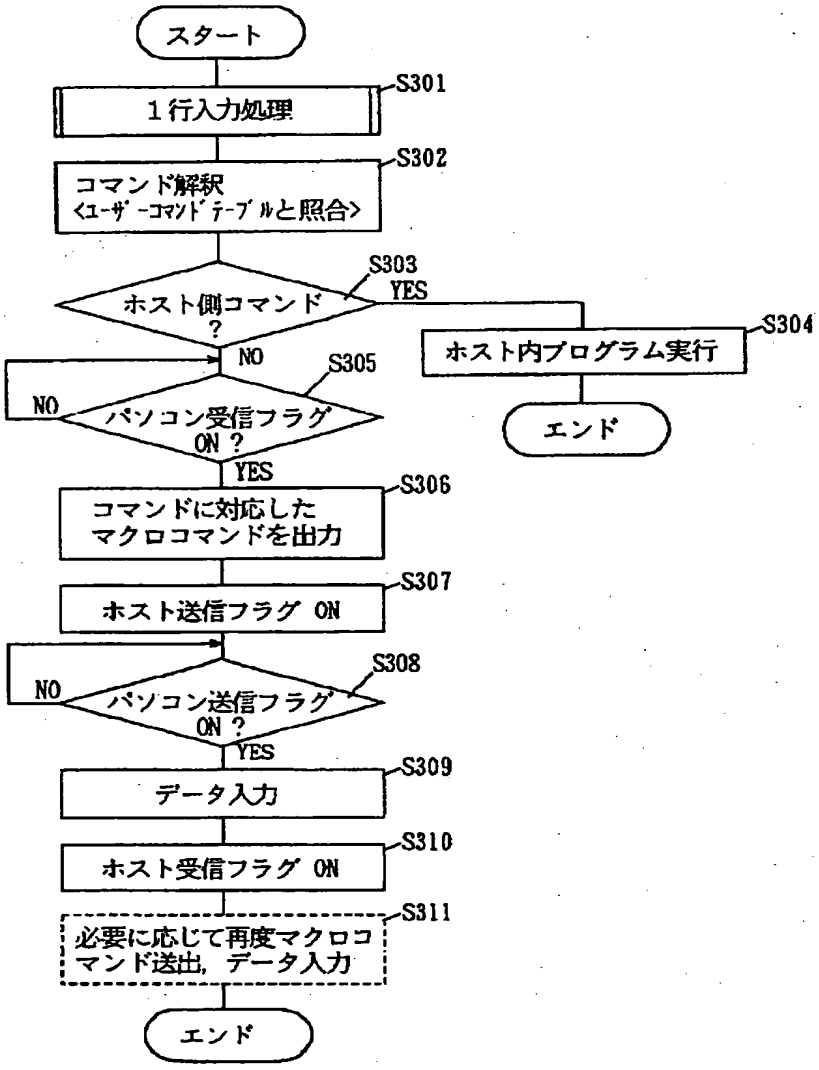
【図７】



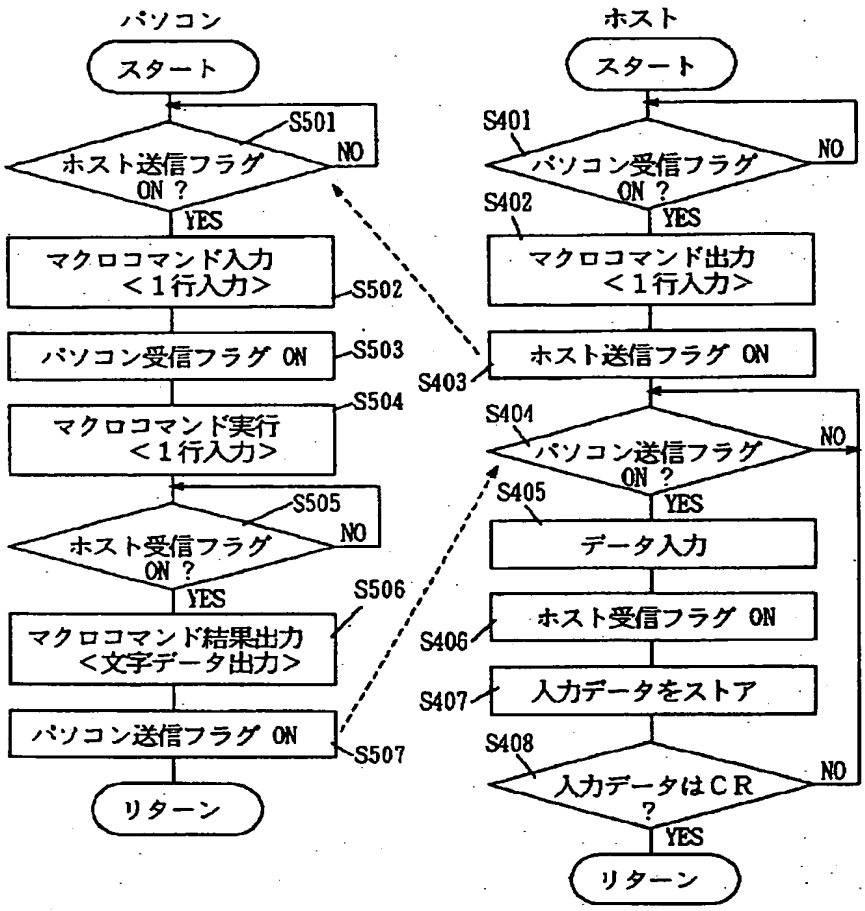
【図１３】



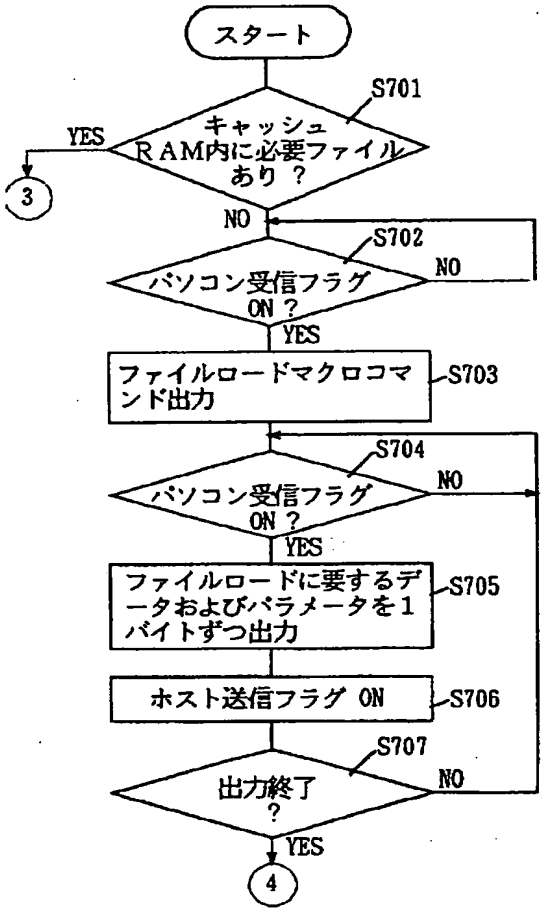
【図８】



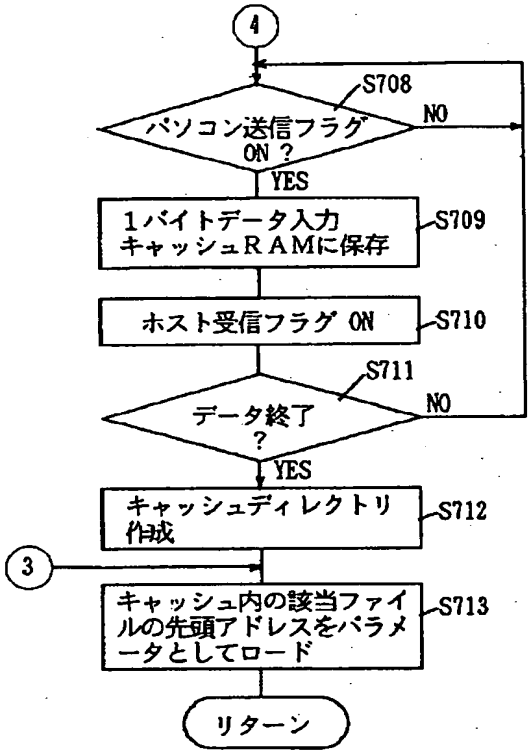
【図９】



【図11】



【図12】



【図１４】